Astronomical Techniques and Instruments, Vol. 1, November 2024, 307–315

Article Open Access

An observation control system for radio telescopes based on Python and C++ languages

Yuxiang Huang[1,2], Longfei Hao[1*], Kejia Lee[3], Wei Dai[4*], Min Wang[1], Zhixuan Li[1], Yonghua Xu[1], Bojun Wang[5], Faxin Shen[1,2]

[1]*Yunnan Observatories*, *Chinese Academy of Sciences*, *Kunming* 650216, *China*

[2]*University of Chinese Academy of Sciences*, *Beijing* 100049, *China*

[3]*Kavli Institute for Astronomy and Astrophysics*, *Peking University*, *Beijing* 100871, *China*

[4]*Computer Technology Application Key Lab of Yunnan Province*, *Kunming University of Science and Technology Kunming* 650050, *China*

[5]*National Astronomical Observatories*, *Chinese Academy of Sciences*, *Beijing* 100101, *China*

*Correspondences: haolongfei@ynao.ac.cn; daiwei@kust.edu.cn

Received: August 8, 2024; Accepted: August 28, 2024; Published Online: September 24, 2024; https://doi.org/10.61977/ati2024040; https://cstr.cn/32083.14.ati2024040

© 2024 Editorial Office of Astronomical Techniques and Instruments, Yunnan Observatories, Chinese Academy of Sciences. This is an open access article under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/)

Citation: Huang, Y. X., Hao, L. F., Lee, K. J., et al. 2024. An observation control system for radio telescopes based on Python and C++ languages. *Astronomical Techniques and Instruments*, **1**(6): 307−315. https://doi.org/10.61977/ati2024040.

Abstract: An observation control system is the foundation to support automatic observations by any radio telescope. Traditional observation control systems are usually coded using a compiled language, which is of higher efficiency compared with interpreted languages. Indeed, observation control systems are usually programmed using the C or C++ languages. However, the high execution efficiency of C/C++ is at the cost of a long development cycle, which is not only time consuming but also requires considerable skills for the developers. The development of computer hardware performance, as well as the optimization of the just-in-time compiler for new interpreted languages such as Python, provides a good balance between execution and development efficiency. In this paper, we introduce the observation control system developed for the Kunming 40-meter radio telescope run by Yunnan Observatories, Chinese Academy of Sciences. The system is developed mainly with the Python language, and we have optimized computationally intensive components with C++. We demonstrate that it is possible to achieve the required functionality and control precision with such a Python-C++ programming paradigm. The performance of the control system is also assessed in this paper, demonstrating that satisfactory pointing accuracy and user experience can be attained.

Keywords: Observation control system; Radio telescopes; Ease of development

1. INTRODUCTION

The Kunming 40-meter radio telescope (KM40m), built in 2006, is a Cassegrain-type antenna with a horizontal mount, designed for data reception and Very Long Baseline Interferometry (VLBI) used in orbital determination for the Chinese Lunar Exploration Program[1]. The telescope later joined international VLBI networks and performed observations for the International VLBI Service for Geodesy and Astrometry, the European VLBI Network, and the Eastern Asia VLBI Network. Pulsars[2] and molecular spectral line observations for scientific research have also been conducted with KM40m. The observation control system (OCS) for KM40m was developed by the 54th Research Institute of the Chinese Electronics Technology Group[3]. In 2018, the Digital Backend System[4], the Digital Base Band Converter (DIBAS)[5], and the sixth-generation VLBI data system Mark6[6] were installed at KM40m, upgrading its observation capabilities. However, the old OCS was unable to control all these new systems. In manual observation mode, extra cost, personnel training, and human power are thus needed. To fully utilize the newly installed observation backends and perform automatic observations, we developed a new OCS based on the Python and C++ languages. Aside from automatic coordination of the backends for observation, the new OCS also improves the antenna pointing accuracy.

For the required execution efficiency, OCSs are usually developed with compiled languages. For instance, the OCS at the Green Bank Telescope (GBT) was developed with the C++ language in 1998[7], an OCS at the Sheshan 25m radio telescope was developed with the C lan-

guage in 2001[8], and the C++ language was chosen for the OCS at the Xinjiang Astronomical Observatory, Chinese Academy of Sciences[9]. Although the C and C++ languages have a high execution efficiency, they have a steep learning curve and usually require more human resources to achieve a given functionality, which makes them inconvenient for rapid iterative development. As a powerful open-source interpretive language with rich third-party libraries, the Python language has a concise and easy-to-read syntax, which is easy to write and debug, making it ideal for rapid development of a lightweight radio telescope OCS. Python has previously been successfully used in this application for observatories such as the Atacama Large Millimeter Array (ALMA)[10] and the Square Kilometer Array (SKA)[11]. Here, we combine the merits of the C++ and Python languages and develop a lightweight radio telescope OCS, named PyOCS. Our development uses the Linux operating system, which has quasi–real-time properties that make it suitable for OCS tasks. The paper is organized as follows: in Section 2, we describe the structure of PyOCS. The performance of the PyOCS is evaluated in Section 3, and concluding remarks are given in Section 4.

2. THE STRUCTURE OF PYOCS

PyOCS is general-purpose telescope control software based on the Python and C++ languages. To date, implemented telescope control functions include pulsar observation and pointing error evaluation. PyOCS adopts a layered architecture in which the system is divided into weakly coupled layers, including the user layer, application layer, and device layer. The system architecture is shown in Fig. 1. Owing to the weak coupling between layers and between internal modules, the system is easy to maintain, update, and expand.

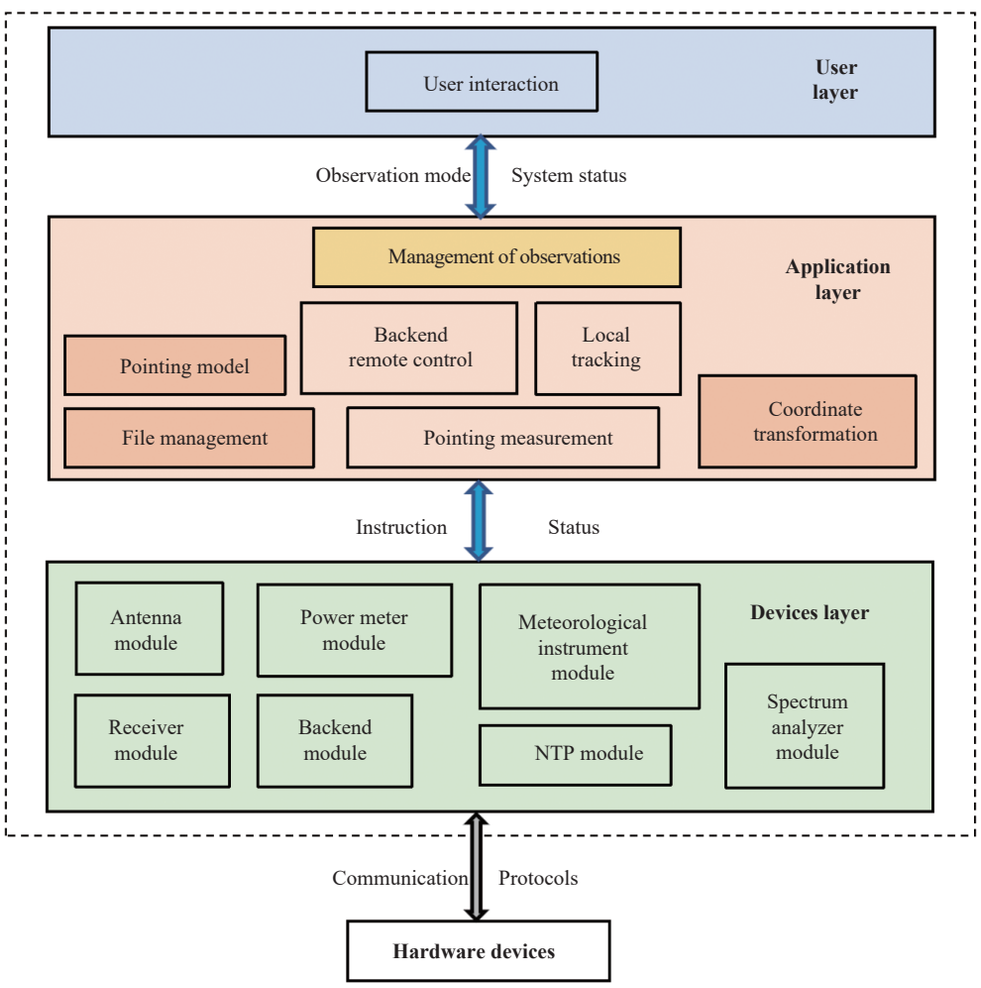

Fig. 1. PyOCS architecture. Three layers are implemented: the user layer, application layer, and device layer. The top layer (i.e., the user layer) allows interaction between the user and PyOCS.

The user layer is designed to allow users to interact with the observation system. It includes an observation system interface and a system monitoring interface. The application layer is composed of function modules required for each observation task, exchanging the commands or system information with the user layer and communicating with the device layer to perform the observation task. The device layer comprises several abstracted hardware mod-

ules. The device layer interacts with the application layer and controls the physical hardware devices via communication protocols (e.g., RS232, TCP/IP).

In general, when a user operates PyOCS, they will interact with the user layer. After choosing an appropriate observation mode, the user layer sends requests to the application layer, which coordinates the modules inside to perform observations. The functional layer also operates abstracted devices via the device layer, which controls the physical hardware. In turn, the functional layer feeds the system status back to the user layer after collecting information from the device layer. The functions and corresponding modules of each layer of PyOCS are explained in the following sections.

2.1. The User Layer

The user layer coordinates functional modules (run as processes or threads) to complete observation tasks and to handle human-computer interactions. The main functions in this layer include: (1) obtaining the configuration information of the telescope and observation task list via the file management module in the application layer; (2) logging the running states of the OCS, observation, and telescope; (3) observation automation by invoking and overseeing the application layer; (4) human interfacing, i.e., displaying the status of each module and receiving operation commands (via command line or graphical interface). To date, we have developed two major functions in the user layer: pointing error measurement and automatic tracking observation according to a task schedule.

Pointing measurements lay down the foundation to assess telescope performance. These measurements are the first step in evaluating the control accuracy, structural precision, and antenna efficiency of a telescope. During the pointing measurement, extragalactic radio sources are scanned, and the received radio power is measured as a function of telescope position, which is usually expressed in equatorial coordinates, i.e., in right ascension (RA) and declination (DEC). A cross trajectory or a spiral trajectory is usually adopted for scanning. The received power is the convolution of the telescope beam shape and source structure, which is the δ-function for the calibrator chosen in the pointing measurement. The beam model can then be fitted to the measured position-power data to obtain the beam shape and pointing deviation at the given sky position. Once the pointing deviations are measured across a sufficient coverage of the sky, the performance of the telescope's pointing can be evaluated, and any deviation can be corrected using the pointing model, provided that deviation is not caused by random fluctuation.

In PyOCS, five major applications are implemented in the application layer to support the pointing measurement function: (1) control of the telescope to follow the given pointing trajectory; (2) interaction with the radio-frequency (RF) power meter (i.e., setup of the meter and taking readings) via the device layer; (3) generation of reports for pointing measurement; (4) storage of scanned data to files; and (5) acquisition of model parameters by fitting to scanned data.

The automatic tracking observation function is probably the most common observation mode used by all radio telescopes. In this function, a pre-determined telescope schedule is loaded into the system so that the telescope will perform observations automatically. In the application layer, the major applications to support automatic tracking observations are: (1) file management applications to load the schedule and log running observations; (2) telescope tracking by controlling targeting and pointing; and (3) control of data-recording backends to record scientific data. PyOCS has already successfully tracked three different types of objects: astronomical sources, solar system bodies, and Earth satellites.

2.2. The Application Layer

The application layer allows the PyOCS components to function correctly. The major applications are: (1) file management module to obtain configuration information for the telescope, to read in observation tasks, and to record the overall PyOCS log; (2) pointing module to point the telescope to the target at the given time; and (3) communication modules to maintain interaction with the device and user layers.

The file management module processes three types of files: configuration files, auxiliary data files, and logs. The configuration files contain site information (longitude, latitude, and altitude) and network parameters (such as IP address, port number, and buffer size) for communication with the device layer. It also handles paths of auxiliary data files, which include pointing model parameters, Earth orientation parameters, solar system ephemeris, calibrator source lists for pointing measurement, output files of pointing measurements, satellite orbits, and observation schedules. The log files are compiled during system operation, where the file management module collects the outputs of the system and creates categories for the log.

Pointing the telescope to the desired position at a given time is the key function of any telescope control system. In PyOCS, it is supported by two sub-components, the coordinate-time transformation component and the pointing correction model component. The coordinate-time transformation component converts the target position at any given time into the local horizontal coordinates of the telescope, as defined by the World Geodetic System. The transformation is implemented with the software package SOFA[12], and the procedure is illustrated in Fig. 2. For astronomical sources, the calculation of this coordinate conversion is well established. PyOCS uses six different time definitions: control computer local time, local atomic clock time, GPS-acquired Coordinated Universal Time (UTC), International Atomic Time (TAI), Terrestrial Time (TT), and Barycentric Dynamic Time (TDB). The conversion between different time definitions is given in Fig. 3. For the observation of celestial objects in the solar system, we calculate their horizontal coordinates using the

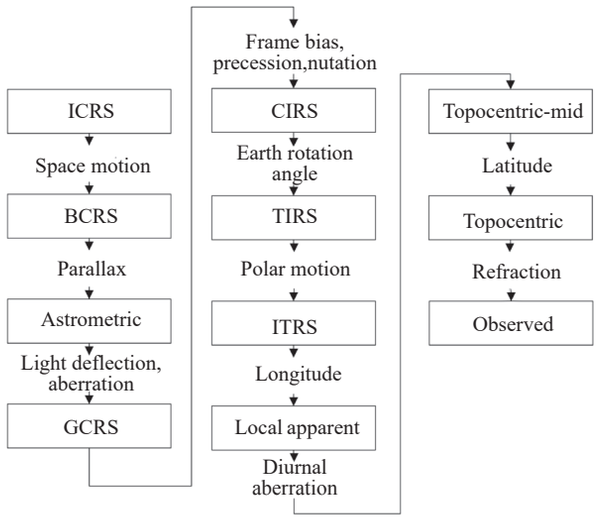

Fig. 2. The transformation process from the international celestial reference system to the local observation coordinate system of the telescope[12].

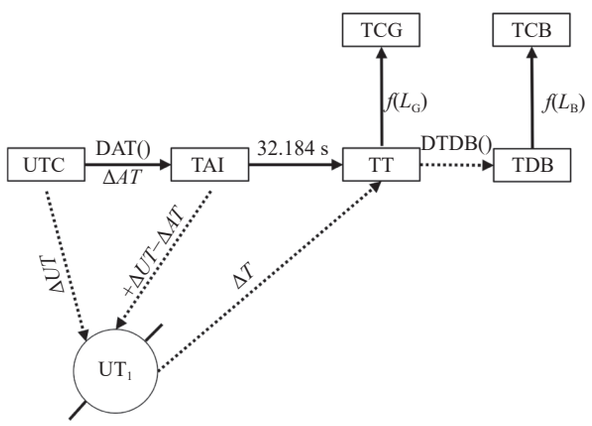

Fig. 3. Transformation between different time definitions[12].

Jet Propulsion Laboratory (JPL) solar system ephemeris. For satellites, we generate real-time horizontal coordinates by interpolating the satellite orbits.

After the horizontal coordinates are computed, the pointing correction model components further convert the ideal horizontal coordinates into physical coordinates used by hardware to drive the telescope, which is defined mostly by rotary encoder readings and includes mechanical errors and gravity deformation[3]. Chinese radio telescopes mostly use two sets of pointing models, namely 8-parameter and 22-parameter models. The models describe the corrections needed for Azimuth (*AZ*) and Elevation (*EL*) as functions of *AZ* and *EL*.

The 8-parameter pointing model is defined as[3]

$$\begin{aligned}\Delta AZ =& C_1 + C_3\tan EL\cos AZ + \\ & C_4\tan EL\sin AZ + \\ & C_5\tan EL - C_6/\cos EL,\end{aligned} \tag{1}$$

and

$$\begin{aligned}\Delta EL =& C_2 + C_3\sin AZ + C_4\cos AZ + \\ & C_7\cos EL - C_8/\tan EL,\end{aligned} \tag{2}$$

where ΔAZ and ΔEL are the deviations of *AZ* and *EL*, respectively; C_1 is the fixed offset of the *AZ* encoder; C_2 is the *EL*-axis component of the alignment error and the fixed offset of the *EL* encoder; C_3 is the east-west tilt coefficient of the *AZ* axis; C_4 is the north-south tilt coefficient of the *AZ* axis; C_5 is the tilt coefficient of the *EL* axis; C_6 is the horizontal component of the alignment error; C_7 is the antenna gravity deformation coefficient; and C_8 is the correction coefficient of atmospheric refraction.

The 22-parameter pointing model[13, 14] is expressed as

$$\begin{aligned}\Delta AZ =& C_1 + C_3\tan EL\cos AZ + \\ & C_4\tan EL\sin AZ + C_5\tan EL - \\ & C_6/\cos EL + C_{12}AZ + C_{13}\cos AZ + \\ & C_{14}\sin AZ + C_{17}\cos 2AZ + C_{18}\sin 2AZ,\end{aligned} \tag{3}$$

and

$$\begin{aligned}\Delta EL =& C_2 - C_3\sin AZ + C_4\cos AZ + \\ & C_7\cos EL + C_8/\tan EL + C_9 EL + \\ & C_{10}\cos EL + C_{11}\sin EL + C_{15}\cos AZ + \\ & C_{16}\sin 2AZ + C_{19}\cos 8EL + C_{20}\sin 8EL + \\ & C_{21}\cos AZ + C_{22}\sin AZ,\end{aligned} \tag{4}$$

where C_1–C_{22} are the parameters of the pointing model. Compared with the 8-parameter model, C_9 to C_{22} are coefficients of higher-order correction terms. Along the *AZ* axis, C_{12} is the coefficient for the *AZ* axis angle encoder scale error, C_{13} is the coefficient for the first-order cosine compensation term in the *AZ* axis direction, C_{14} is the coefficient for the first-order sine compensation term in the *AZ* axis direction, C_{17} is the coefficient for the second-order cosine compensation term in the *AZ* axis direction, and C_{18} is the coefficient for the second-order sine compensation term in the *AZ* axis direction. Along the *EL* axis, C_9 is the coefficient for the elevation axis angle encoder scale error, C_{10} is the coefficient for the gravity deformation compensation term, C_{11} is the coefficient for the first-order sine compensation term in the *EL* axis direction, C_{15} is the coefficient for the second-order cosine compensation term in the *AZ* axis direction, C_{16} is the coefficient for the second-order sine term in the *AZ* axis direction, C_{19} is the coefficient for the eighth-order cosine compensation term in the *EL* axis direction, C_{20} is the coefficient for the second-order sine term in the *EL* axis direction, C_{21} is the coefficient for the *AZ* axis east-west tilt compensation term, and C_{22} is the coefficient for the *AZ* axis north-south tilt compensation term. Theoretically, the 22-parameter model can improve pointing accuracy. Among the 22 parameters, C_7 and C_{10}, C_3 and C_{22}, and C_4 and C_{21} are degenerate terms, and one of them is fixed

during fitting. We find that in the fitting for actual pointing parameters, the parameters C_9−C_{11} have a large fitting error, i.e., they are not well constrained. We remove these three terms for practical application[14]. PyOCS can switch between multiple pointing models by changing the configuration files. Here, we focus only on the 22-parameter pointing model, for simplicity.

2.3. The Device Layer

The device layer contains abstractions of hardware devices, with each type of physical hardware corresponding to an abstract device class. For example, the antenna class encapsulates pre-defined control commands and state properties necessary for pointing operations. One can extend to a new antenna by inheriting the antenna class and adding/overloading any necessary functions. The device layer makes sure that no modification is needed for the application layer when adding new hardware to the system if the hardware base class has been implemented. In the abstract device class, we encapsulate a few common hardware interfaces, such as an RS232 serial interface and a TCP/IP interface to support easier implementation of individual hardware devices. At present, we have implemented device classes for the antenna, the supporting devices (e.g. weather station and RF power meter), and data recording backends such as Pulsar Digital Filter Banks (PDFB)[15], Burst Emission Automatic Roger (BEAR)[16], and DIBAS.

3. EVALUATING THE PERFORMANCE OF PYOCS

3.1. The Pointing Accuracy Evaluation

Pointing accuracy is one of the most critical specifications for observation control software. The design pointing accuracy of KM40m is 30″, and acceptance testing of KM40m recorded a pointing accuracy of 28″ after applying the pointing correctional model[3]. In this section, we explain the method to measure the raw pointing deviation of the telescope and the techniques to derive the pointing model, and then evaluate the telescope's pointing accuracy after applying the pointing model correction.

We choose the X band (8−9 GHz) to perform the pointing deviation measurement. This is the highest possible working frequency band of KM40m, with which the telescope beam width is minimal and the design specifications of pointing accuracy are 21″ in the elevation (*EL*) direction and 21/cos(*EL*)″ in the azimuth (*AZ*) direction. We measure the pointing deviations by scanning a set of radio sources distributed across the sky. With an *EL* limit of ⩾ 8°, KM40m can scan all radio sources with *DEC* ⩾ −57°. To obtain a sufficiently high signal-to-noise ratio, radio sources with X-band flux S ⩾ 2.0 Jy were selected for scanning. To get good coverage over the *AZ* and *EL* parameter space, a set of suitable radio sources was selected at every 10° interval in the *DEC* direction. The radio sources used[17] are listed in Table 1.

To measure the pointing deviation, we perform pointing scanning, in which the received power of the telescope is recorded as a function of telescope position around bright celestial sources with known positions. Then we can find the telescope pointing deviation by measuring the offset between the position predicted by the telescope control system and the position, at which we receive maximum power. The scans use cross-shaped paths, i.e., tracking toward the *AZ*+ (right) direction, followed by the *AZ*− (left), *EL*− (lower), and *EL*+ (upper) directions, with a range of [−0.2°, +0.2°]. Telescope output radio power is recorded at a total of 480 positions, uniformly distributed along the four paths, with each scan taking approximately 5 minutes. The scans in this work are shown in Fig. 4. As will be seen in Fig. 5, the raw pointing deviation is at the level of degree. These scans use

Table 1. Radio sources used in the pointing measurement

Source	RA	DEC	Source	RA	DEC
3C48	01:37:41.27	33:09:35.7	3C145	05:35:16.4	−05:32:23.0
3C84	03:19:48.16	41:30:42.1	3C161	06:27:10.1	−05:53:05.0
3C123	04:37:04.17	29:40:15.1	3C218	09:18:05.7	−12:05:44.0
3C144	05:34:32.00	22:00:58.00	3C279	12:56:11.1	−05:47:22.0
3C147	05:42:36.14	49:51:07.2	3C345	16:51:08.1	04:59:33.0
3C273B	12:29:06.70	02:03:08.60	3C353	17:20:28.1	−00:58:47.0
3C274	12:30:49.42	12:23:28.0	NRAO530	17:33:02.7	−13:04:50.0
3C295	14:11:20.65	52:12:09.1	4C39.23	08:24:55.483	39:16:41.91
3C348	16:51:08.20	04:59:33.0	NGC7027	21:07:01.59	42:14:10.186
3C380	18:29:31.72	48:44:47.0	1830−210	18:33:39.915	−21:03:40.05
3C405	19:59:28.4	40:44:02.0	0208−512	02:10:46.2	−51:01:01.9
3C461	23:23:24.8	58:48:59.0	1921−293	19:24:51.056	−29:14:30.121
3C286	13:31:08.288	30:30:32.96	0854+2006	08:54:48.8749	20:06:30.640
3C138	05:21:09.886	16:38:22.05	1058+0133	10:58:29.6052	01:33:58.823
Orion	05:35:16.47	05:23:22.84	2136+0041	21:36:38.5863	00:41:54.213
DR21	20:39:01.1	42:19:43.0	2253+1608	22:53:57.7479	16:08:53.560

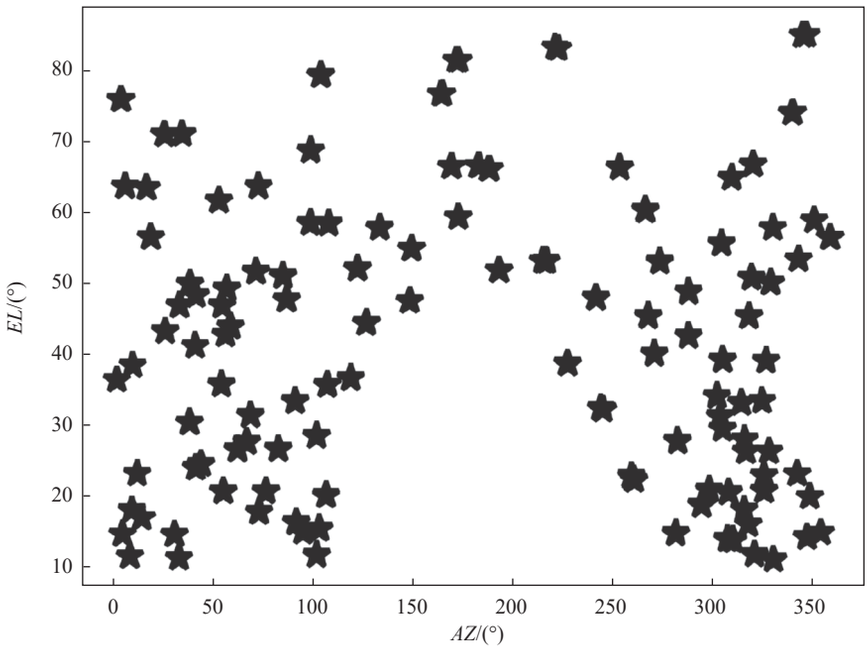

Fig. 4. The all-sky coverage of scanning points.

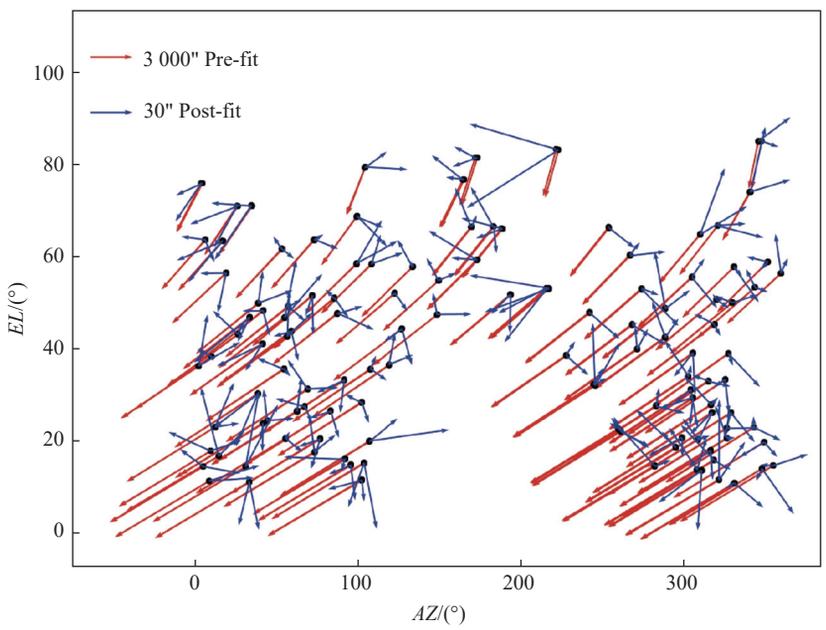

Fig. 5. Vector field of pointing errors. The arrows indicate the pointing error vector. The data before and after fitting are shown in red and blue, respectively.

the old pointing model, measured using the acceptance evaluation of 2006. Outliers are rejected with pointing deviations larger than 0.3° after correction with the old pointing model, i.e., we remove data points nine half-beam widths away, which are mainly caused by radio interference. The true pointing deviations are then computed by subtracting the pointing model corrections.

After scanning, we fit the beam model to the position-RF power datasets to give the pointing deviation and beam parameters. We model the RF power as a two-dimensional Gaussian function with respect to pointing deviation and add a third-order polynomial[18] to describe the baseline variation of RF power during scanning. The fitting is performed using the standard Bayesian method, implemented with the software package multinest, which can effectively converge to a global optimal solution[19].

We denote the i-th measurement with subscript i and $i \in [1, N]$. The coordinates of each scan are AZ_i and EL_i, and the corresponding pointing deviations are ΔAZ_i and ΔEL_i. We denote the measurement errors of the pointing deviation as δAZ_i and δEL_i. The errors are computed using the Bayesian method and contain both statistical errors and systematics because of the correlation induced by baseline variation. The measured field of the pointing

deviation is shown in Fig. 5, showing a systematic trend in the pointing deviation before applying the pointing model, which is on the order of degrees. The pointing model is then derived by minimizing the weighted pointing error, which is expressed as

$$\delta_{\text{point}} = \sqrt{\frac{\sum_i \left[\frac{(\Delta AZ_i \cos EL_i)^2 + \Delta EL_i^2}{\sigma_{AZ_i}^2 (\Delta AZ_i \cos EL_i)^2 + \sigma_{EL_i}^2 \Delta EL_i^2} \right]}{\sum_i \left[\frac{1}{\sigma_{AZ_i}^2 (\Delta AZ_i \cos EL_i)^2 + \sigma_{EL_i}^2 \Delta EL_i^2} \right]}}. \quad (5)$$

After applying the new pointing model, the residual δ_{point} is the accuracy of the pointing system. The measured value is $\delta_{\text{point}} \approx 25''$, which is slightly better than the previous accuracy of 28″. The 22 parameters of the pointing model are shown in Table 2.

After applying the new pointing model, we performed another scan to verify the precision. The result is shown in Fig. 5. Using the new model, the deviations have no system trends, and the level is reduced to the order of 10″. The statistical properties of the post-correction pointing deviations are shown in Fig. 6. The residual pointing deviations roughly follow a two-dimensional Gaussian distribution, the standard deviations of which are 18″ in the *AZ* and *EL* directions.

Currently, we have not determined the cause of the residual pointing error. Temperature fluctuation, wind pressure, mechanical clearance, servo motor accuracy, and surface deformation are all possible causes, and further investigation is a topic for future research.

3.2. The Evaluation of Antenna Tracking Accuracy

We further examine the tracking characteristics of the telescope by tracking a radio calibrator source and checking the signal-to-noise ratio of the received signal as a function of observation time. The calibrator 3C147 was chosen, of which the angular diameter is much smaller than the antenna beam (details are given in Table 3). In the tracking process, any accumulated pointing deviation leads to a decrease in the signal-to-noise ratio. We observed 3C147 for approximately 90 minutes, during which its elevation fell in the range of $EL \in [47°, 60°]$. We used the ON-OFF observation to correct the influence of background variation and gain instability, i.e., during an observation, the telescope is pointed to the source for 5 minutes (ON state) and then shifted by 2° in the *AZ* direction (OFF state) for 5 minutes. The flux of the radio source is proportional to the difference between the measured powers of ON and OFF states. Owing to the instability of the system gain, the measured power of both ON and OFF states will vary slowly. However, if the system tracking performance has no long-term drift and the system is stable over one ON-OFF cycle, the power ratio $\eta = (P_{\text{ON}} - P_{\text{OFF}})/P_{\text{OFF}}$ will be constant. η is also the ratio

Table 2. The 22 parameters of the pointing model

Parameter	Value/(°)	Parameter	Value/(°)
C_1	−1.878388	C_{12}	0.002756
C_2	−1.150070	C_{13}	0.007051
C_3	−0.004128	C_{14}	0.003761
C_4	−0.003461	C_{15}	−0.007191
C_5	0.022519	C_{16}	0.006641
C_6	0.055187	C_{17}	−0.011970
C_7	−0.015006	C_{18}	−0.000460
C_8	−0.006663	C_{19}	0.003514
C_9	0.039520	C_{20}	0.002229
C_{10}	−0.009590	C_{21}	0.000657
C_{11}	0.1162899	C_{22}	−0.005438

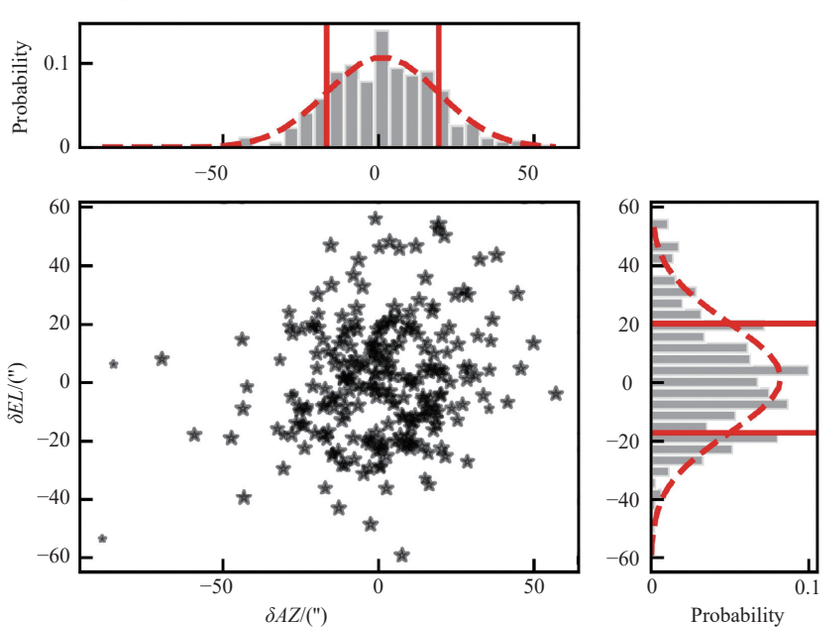

Fig. 6. Post-correction pointing deviation. Top and right panels are the histograms of the *AZ* and *EL* deviations, respectively. The red curve is a Gaussian probability distribution matching the average and standard deviation of the data. Solid red lines indicate standard deviations.

between the calibrator flux and the equivalent flux of the system temperature. We can judge whether the tracking is stable by the stability of that ratio. The results of the 3C147 observation are shown in Fig. 7. Although both the ON and OFF power fluctuate, the power ratio η is approximately constant without any long-term drift, which is what we expect for a pointing accuracy of 25″, given that the X-band full-width-at-half-maximum beam size is approximately 4′.

Table 3. Parameters of 3C147 at 8.1 GHz[20]

Source	RA	Dec	Flux/Jy	Angular size/(")
3C147	05:42:36.14	49:51:07.23	4.83	0.7

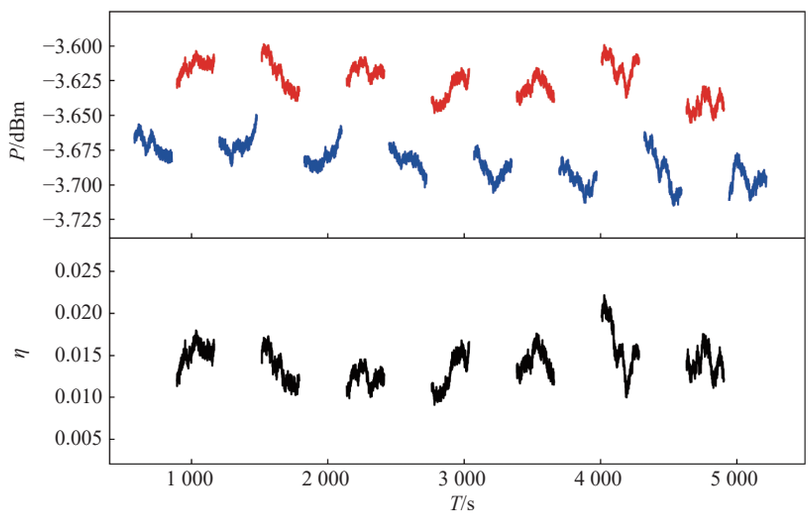

Fig. 7. Top: The recorded power of 3C147 after baseline elimination. Bottom: The power ratio η, where we estimate the corresponding OFF power by interpolating the adjacent measurements.

4. CONCLUSIONS

This paper introduces the telescope control software system PyOCS, developed for KM40m. PyOCS is implemented in the Python and C++ languages, with the framework and modules mainly developed using Python and computationally intensive components (i.e., coordinate and time transformation) written in C++. PyOCS is demonstrably capable of the basic functions required for radio telescope observation: pointing measurement and calibration, and tracking observation of celestial sources and artificial satellites. After pointing calibration, the accuracy of the PyOCS controlling KM40m meets the observation requirements at 25″ accuracy, which is comparable to that of the previous control system. An area of substantial improvement using PyOCS, compared with the previous system, is that we now have a framework where new instruments and data recording backends can be easily integrated for automatic observations.

The control accuracy of PyOCS with KM40m has been measured using pointing scanning. We further verify the stability using a 1.5-hour tracking observation towards the calibration radio source 3C147. At present, the PyOCS is in practical use at KM40m and has been supporting daily observation tasks. In future work, we will develop an OCS for the Jingdong 120-m pulsar radio telescope based on PyOCS.

ACKNOWLEDGEMENTS

This work was funded by the National SKA Program of China (2020SKA0120100), the Special Project of Foreign Science and Technology Cooperation of Yunnan Provincial Science and Technology Department (202003AD150010), and the National Natural Science Foundation of China (12073076, 12173087, 12041303 and 12063003), the Foundation of the Chinese Academy of Sciences (Light of West China Program), the CAS-MPG LEGACY Project and the Max-Planck Partner Group.

AUTHOR CONTRIBUTIONS

Yuxiang Huang, Longfei Hao, Kejia Lee, and Wei Dai provided the research topic and conceived ideas. They designed, wrote, and tested control programs. They also wrote the paper. Min Wang, Zhixuan Li, Yonghua Xu, Bojun Wang, and Faxin Shen participated in discussions on program writing issues, data processing and analysis, and revised the paper. All authors have read and approved the final manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

[1] Zhang, H. B., Mao, P. F., Wang, M, et al. 2008. The 40m radio telescope. *Astronomical Research & Technology*, **5**(2): 187−191. (in Chinese)

[2] Xu, Y. H., Lee, K. J., Hao, L. F., et al. 2018. Interstellar

scintillation observations for PSR B0355+54. *Monthly Notices of the Royal Astronomical Society*, **476**(4): 5579−5590.
[3] Gao, G. N. 2007. Operation test and analysis of antenna Control system and antenna pointing error correction of the 40-meter radio telescope for Yunnan Observatories, CAS. Master's Thesis. Yunnan Observatories, CAS.
[4] Ford, J. M., Prestage, R. M., Bloss, M. 2014. Experiences with the design and construction of wideband spectral line and pulsar instrumentation with CASPER hardware and software: the digital backend system. In Proceeding of SPIE. 9152:915218.
[5] Tuccari, G., Buttaccio, S., Nicotra, G., et al. 2009. DBBC.2 Backend System: status report. In Proceedings of the 19th European VLBI for Geodesy and Astrometry Working Meeting.
[6] Guo, S. G., Zhu, R. J., Zheng, W. M., et al. 2017. The progress of VLBI data acquisition system-Mark6 and related technologies. *Astronomical Research & Technology*, **14**(3): 281−287. (in Chinese)
[7] Fisher, J. R. 1998. Object-oriented experiences with GBT monitor and control. In Proceedings of Astronomical Data Analysis Software and Systems VII.
[8] Xue, Z. H. 2001. S2FS software for S2 terminal of 25m radio telescope. Annals of Shanghai Observatory Academia Sinca, 22: 119–121. (in chinese)
[9] Hu, Y., Aili, Y., Zhao, R. B. 2008. The design and realization of radio telescope control software in Linux operating system. *Astronomical Research & Technology*, **5**(2): 199−205. (in Chinese)
[10] Farris, A., Marson, R., Kern, J. 2005. The ALMA telescope control system. In Proceedings of the 10th International Conference on Accelerator and Large Experimental Physics Control Systems.
[11] Williams, S. J., Bridger, A., Chaudhuri, S. R., et al. 2016. The SKA observation control system. In Proceedings of SPIE. 9913:99132L.
[12] Hohenkerk, C. Y. 2011. SOFA—a status report, review and look to the future. In Proeedings of the Journées 2010 Systèmes de Référene Spatio-temporels.
[13] Himwich, E. 2000. Introduction to the field system for non-users. In Proceedings of International VLBI Service for Geodesy and Astrometry 2000 General Meeting.
[14] Li, J. D., Zhao, D. X., Liu, C., et al. 2020. Analysis of pointing scan data of the Sheshan 13m radio telescope. *Geomatics and Information Science of Wuhan University*, **45**(2): 159−166. (in Chinese)
[15] Qian, M. F., Wang, N., Liu, Z. Y., et al. 2011. Flux density measurements of 23 pulsars. In Proceedings of 2011 International Conference on Electronics and Optoelectronics.
[16] Men, Y. P., Luo, R., Chen, M. Z., et al. 2009. Piggyback search for fast radio bursts using Nanshan 26 m and Kunming 40 m radio telescopes – I. Observing and data analysis systems, discovery of a mysterious peryton. *Monthly Notices of the Royal Astronomical Society*, **488**(3): 3957−3971.
[17] Massardi, M., Ekers, R. D., Murphy, T., et al. 2008. The Australia telescope 20-GHz (AT20G) survey: the bright source sample. *Monthly Notices of the Royal Astronomical Society*, **384**: 775−802.
[18] Yun, L. F., Wang, J. Q., Zhao, R. B., et al. 2015. Pointing model establishment of TM65m radio telescope. *Acta Astronomica Sinica*, **56**(2): 165−177. (in Chinese)
[19] Feroz, F., Hobson, M. P., Bridges, M. 2009. MULTINEST: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Monthly Notices of the Royal Astronomical Society*, **398**(4): 1601−1614.
[20] Perley, R. A., Butler, B. J. 2017. An accurate flux density scale from 50 MHz to 50 GHz. *The Astrophysical Journal Supplement Series*, **230**(7): 1−18.